집중교육1

Easy-follow 프로젝트

중간 프로젝트 보고서

2조 (주황팀)

201720707 나용성

201921085 곽수정

201620985 전성주

201620952 백승수

**목차**

1. 프로젝트 소개
2. 프로젝트 구상

2.1 앞차 기능 구상

2.1.1 앞차 기본 기능 구상

2.1.2 앞차 추가 기능 구상

2.2 뒷차 기능 구상

2.2.1 뒷차 기본 기능 구상

1. 요구사항 식별

3.1 앞차 요구사항

3.1.1 앞차 기본 기능 요구사항

3.1.2 앞차 추가 기능 요구사항

3.2 뒷차 요구사항

3.2.1 뒷차 기본 기능 요구사항

1. 요구사항 분석

4.1 자동차의 움직임에 대한 이해

4.2 센서에 대한 이해

4.2.1 초음파 센서

4.2.2 조도 센서

4.2.2.1 밝기 측정 테스트

4.3 앞차 기능 분석

4.3.1 앞차 뒷바퀴 회전 속도

4.3.2 앞차 조향장치 분석

4.4 뒷차 기능 분석

4.4.1 선행 차량 식별

4.4.2 전진, 후진 및 정지

4.4.3 좌우 방향 전환

1. 설계

5.1 앞차 설계

5.1.1 앞차 TASK 식별

5.1.2 BtReceiver

5.1.3 DistSensing

5.1.4 Operator

5.2 뒷차 설계

5.2.1 Task 설계

5.2.1.1 sensing (basic)

5.2.1.2 drive (Extended)

5.2.1.3 back (Extended)

5.2.1.4 stop (Extended)

5.2.1.5 steering (basic)

5.2.2 Priority

5.2.3 Event

5.2.4 Resource

5.2.5 Alarm 설계

5.2.6 아쉬운점

1. 프로젝트 소개

본 프로젝트는 크게 선행 차량과 후행 차량, 총 2대의 차량을 목적에 알맞게 구현하는 것이다. 선행 차량은 블루투스 연결을 통해 사용자의 입력에 반응하며, 그에 따라 전후 좌우 이동 및 추가 기능을 수행하는 차량이다. 후행 차량은 사용자의 입력없이 차량 스스로 선행 차량을 식별하고, 식별된 차량을 추적하며 이동하는 차량이다.

2. 프로젝트 구상

2.1 앞차 기능 구상

2.1.1 앞차 기본 기능 구상

앞차는 기본적으로 과제에서 주어진 목표인 핸드폰과 블루투스로 연결되어 핸드폰의 앱을 통해 전후좌우로의 이동과 속도조절, 급브레이크와 서서히 감속하는 제동을 수행할 수 있는 조향장치가 포함된 4륜구동 모형자동차로 만들어져야 한다.

2.1.2 앞차 추가 기능 구상

기본기능에 더해 추가로 운전자의 편의와 안전을 위해 부분적 자율주행을 구현하기 위해 주행보조 기능으로 전방 장애물과의 거리를 초음파센서로 식별하여 미리 감속하고 일정거리 이하에선 제동하는 충돌방지모드와 조도센서를 통해 차량이 차선을 이탈하려할 때 이를 식별하고 차선이탈이 식별됐을 시 차량의 진행방향을 조정하는 차선이탈 방지모드를 구현하고 이를 앱의 추가확장 버튼을 누른 시간에 따라 개별적으로 활성화, 비활성화할 수 있도록 개발하고자 하였다.

2.2 뒷차 기능 구상

2.2.1 뒷차 기본 기능 구상

뒷차량은 선행 차량을 안정적으로 따라가는 것이 목적이다. 이를 위해 첫번째로 선행 차량과의 거리를 정확하게 인식해야할 필요가 있다. 또한 선행 차량은 전진 및 후진, 정지, 고속 및 저속, 좌우 방향 전환을 통해 이동 경로가 변화하기 때문에 뒷차는 이러한 변화를 인식하고 그에 맞춰 이동해야할 필요가 있다. 따라서 뒷차도 앞차량의 이동 방식과 동일하게 전 후진, 정지, 고속 저속 운행, 좌우 방향 전환 기능이 필요하다고 생각하였다.

3. 요구사항 식별

3.1 앞차 요구사항

3.1.2 앞차 기본 기능 요구사항

* 앞차는 핸드폰과 블루투스로 연결되어 앱을통한 운전자의 입력을 수신할 수 있어야 하며 연결되지 않은 동안은 가동하지 않아야 한다.
* 앞차는 운전자가 원하는 속도로 전진 후진을 하도록 뒷바퀴의 회전 방향과 속도를 변경할 수 있어야 한다.
* 앞차는 중립, 좌회전, 우회전 중 운전자가 원하는 방향으로 진행할 수 있도록 조향장치를 담당하는 모터의 각도를 정해진 각도까지만 회전시킬 수 있어야 한다.
* 앞차는 운전자의 입력에 따라 설정된 브레이크 모드로 급제동하거나 서서히 제동할 수 있어야한다.

3.1.3 앞차 추가 기능 요구사항

* 앞차는 운전자가 추가확장 버튼을 누른 시간에 따라 충돌방지모드, 차선이탈방지모드 중 사용자가 원하는 모드를 활성화/비활성화 할 수 있어야 한다.
* 앞차는 충돌방지모드가 활성화 되어있는 동안 초음파 센서를 통해 전방 장애물과의 거리를 측정하여 장애물과 가까워지면 미리 감속하고 일정거리 이하에서는 제동할 수 있어야 한다.
* 앞차는 차선이탈방지모드가 활성화 되어있는 동안 조도센서를 통해 차선을 감지하여 차선을 이탈하는것을 감지하고 조향장치를 조정하여 차선이탈을 방지할 수 있어야 한다.

3.2 뒷차 요구사항

3.2.1 뒷차 기본 기능 요구사항

2.2.1 에서 구상한 기본 기능을 수행하기 위해선 아래와 같은 요구사항을 수행 해야 한다.

* 뒷차는 앞차량을 식별할 수 있어야 하고 두 차량 사이의 거리를 측정할 수 있어야 한다.
* 뒷차는 앞차량의 진행방향을 식별하여 다음 진행방향을 결정할 수 있어야 한다.
* 뒷차는 주행중인 앞차량을 따라 주행할 수 있어야 한다.
* 뒷차는 주행중인 앞차량의 거리가 일정거리 이상 차이가 나면 속도를 조절하여 일정 거리를 유지 할 수 있어야 한다.
* 뒷차는 주행 혹은 정지중인 앞차와의 추돌을 방지하기 위해 거리가 일정거리 이상 가까워 지면 정지 하거나 후진 할 수 있어야 한다.
* 앞차량의 좌우 방향 전환에 따라, 뒷차량의 스티어링을 통해서 좌우 방향 전환을 할 수 있어야 한다.

4. 요구사항 분석

4.1 자동차의 움직임에 대한 이해

자동차의 기능을 분석하기 전에 자동차의 움직임에 대해 이해를 할 필요가 있다. 본 프로젝트에서 구현하고자 하는 자동차는 조향장치가 있는 후륜구동 4륜차이다. 후륜구동이기 때문에 자동차의 동력을 뒷바퀴가 담당하고, 뒷바퀴에 설정된 속도가 자동차의 속도가 되며 뒷바퀴의 회전 방향에 따라 전진 또는 후진의 방향이 결정된다. 자동차의 진행 각도 변경은 조향 장치로서 동작하는 앞바퀴 모터의 각도에 따라 결정된다. 위와 같은 후륜 구동 4륜 자동차의 작동 원리를 이해함에 따라 자동차의 움직임을 결정하는 요소를 다음과 같이 식별할 수 있었다.

* 뒷바퀴의 회전 방향
* 뒷바퀴의 회전 속도
* 조향 장치의 각도

선행 차량은 사용자의 입력에 따라서 세 요소를 변경하여 차량의 이동을 구현할 수 있고, 후행 차량은 센서 인식에 따라 선행 차량과 같은 방식으로 차량의 이동을 구현할 수 있다.

4.2. 센서에 대한 이해

4.2.1. 초음파 센서

초음파 센서는 초음파가 발사되고 돌아오는 시간을 측정하여 물체와의 거리를 측정할 수 있는 센서이다. 앞차의 추가 기능인 충돌 방지 모드, 뒤차의 기본 기능인 앞차 식별을 구현하기 위해서 초음파 센서의 특성이 적합하다고 판단하였다. 따라서 초음파 센서를 사용하기 위해 여러 테스트를 진행하였고, 다음과 같은 초음파 센서의 특징을 알 수 있었다.

* 초음파의 발사 후 되돌아오기까지의 시간이 필요하기 때문에 초음파 센서를 통한 2회 이상의 측정간에는 최소 시간간격이 필요하며 그 시간간격은 최소한 센서별로 50ms가 적당하다.
* 발사된 초음파가 적중된 면이 정면이 아닌 빗면이거나 면이 고르지 못한 경우 예상치 못한 각도로 반사되어 측정이 실패되는 경우 초음파 센서가 측정 가능한 최대 거리가 반환된다.
* 각 초음파 센서들은 동일한 주파수를 이용하기 때문에 두 센서 사이의 거리가 너무 가까운 경우 서로 간섭이 발생하는 것을 알 수 있었다.

특히 두번째 특징의 경우 꽤나 빈번히 발생하는 상황이므로 우리는 수업에서 노이즈를 필터링할 수 있는 방법을 몇가지 배웠고 그 중 7회 측정하여 측정값 중 중간값을 사용하는 방식을 사용하기로 했다.

4.2.2. 조도센서

조도센서는 밝기를 감지하는 센서이다. 제공된 조도센서는 특정 API 를 통해서 붉은 빛을 발생시킨다. 이러한 기본적인 특성을 제외한 또 다른 조도센서의 특성을 이해하기 위해 하단에 서술한 테스트를 진행해보았다.

4.2.2.1 밝기 측정 테스트

첫번째 테스트 : 우리는 조도센서가 밝기를 측정할 때 측정값의 변화를 알기 위해 조도센서를 active시키고 불이 밝은 환경과 어두운 환경에 둔 결과 조도센서는 어두운 환경에서 높은 값을 반환하고 밝을수록 낮은 값을 반환했다.

두번째 테스트 : 빛은 어두운 색의 물체에 부딪혔을 때 흡수되고 밝은 물체에 부딪혔을 땐 반사되고 조도센서는 active시켰을 때 붉은 빛이 나오므로 조도센서를 물체 가까이 갖다대면 밝은 물체는 빛을 많이 반사하여 조도센서에 빛이 많이 감지되고 어두운 물체는 빛을 흡수하여 조도센서에 빛이 덜 감지될 것으로 예상하여 하얀 A4용지에 검은 부분을 일부 출력하여 조도센서를 하얀 부분에서 검은 부분으로 이동해가며 측정값의 변화를 살펴보았다.

두번째 테스트 결과 : 조도센서는 하얀 부분에선 420±10정도의 값을 반환했고 검은 부분에선 650±10의 값을 반환했으며 하얀 부분과 검은부분의 경계에 진입한 부분부터 450이상의 값을 반환하며 검은 부분으로 이동할 수록 측정값은 높아져갔다.

그러므로 우리는 차선의 경계에 대한 기준을 정하고 조도센서의 측정값이 이를 벗어나는지를 지속적으로 측정하여 차선이탈을 감지할 수 있을것이다.

4.3 앞차 기능 분석

4.3.1 앞차 뒷바퀴 회전속도

뒷바퀴는 앞에서 차량의 전진 또는 후진 방향과 차량의 속도를 결정짓는다는것을 알았다. 구현할 기능 중 앞의 요소를 결정짓는 요소는 다음과 같다.

* 운전자의 핸드폰 앱을 통해 입력된 전진, 또는 후진 입력, 운전자가 입력한 속도, 브레이크 여부
* 추가기능인 충돌방지를 위해 측정된 거리

위의 요소 중 거리의 경우 50ms주기로 7회 측정 후 유효한 값을 산출하기 때문에 속도에 반영되려면 350ms의 주기가 필요하며 운전자의 입력은 매우 빠른 주기로 확인할 수 있다. 그러므로 운전자의 입력을 수시로 저장하는 기능과 거리를 측정하는 기능을 분리한 뒤 거리를 측정하는 기능에서 저장된 운전자의 입력을 고려해 뒷바퀴의 회전속도를 결정하는것이 좋을 것이라 생각했다. 그러나 브레이크는 위급한 상황에 사용이 많이되는 동작이기 때문에 브레이크의 입력이 운전자의 입력을 저장하는 기능에서 식별될 시 운전자의 입력을 저장하는것과 추가로 속력에 직접 브레이크 값을 반영하도록 해야 할 것이다.

그리고 앞차에 초음파 센서는 전방에만 부착되어있기 때문에 전진 시에만 측정된 거리에 따른 속도 감속이 적용되면 되며 감속이 시작되는 구간은 고속에서가 저속에서보다 제동거리가 길고 거리가 측정되는데 걸리는 시간도 필요하기 때문에 감속이 시작되는 구간도 더 길게 책정해야 할 것이다. 우리는 고속의 최대속도를 60, 저속은 그 절반인 30으로 감속이 시작되는 거리는 고속에서 100, 저속에선 80으로 결정했다. 그러므로 충돌방지모드가 활성화 되었을 때 전진속도는 다음의 그래프와 같이 변하게 된다.

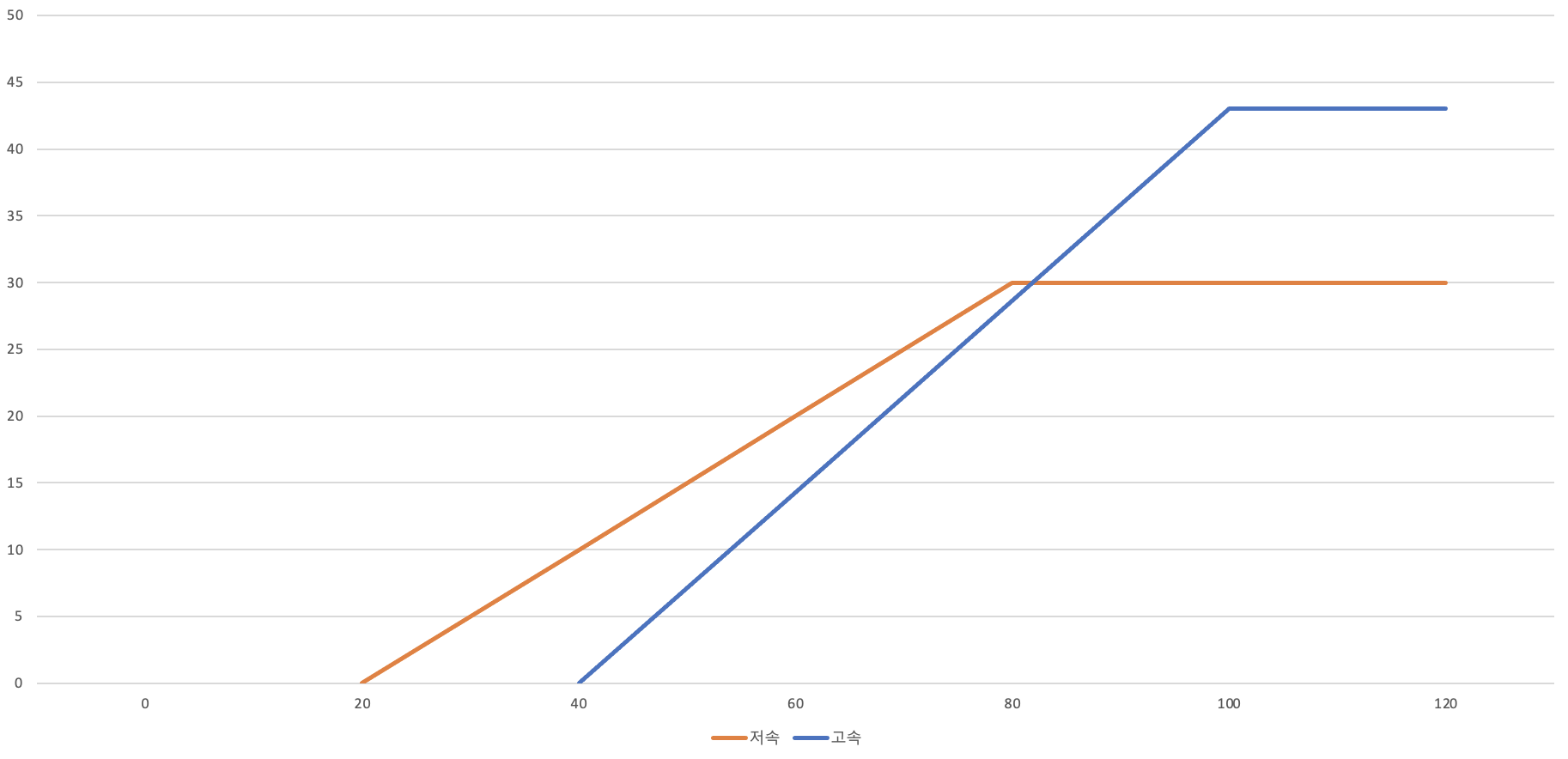


그림 4-1 측정 거리와 속도 설정에 따른 전진속도 변화 그래프

충돌방지모드가 비활성화 되었거나 후진인 경우엔 고속일 때 60, 저속일 때 30으로 회전속도를 결정하고 전진일 때 -1을 곱해 방향을 서로달리 함으로써 뒷바퀴를 제어할 수 있을것이다.

4.3.2 앞차 조향장치 분석

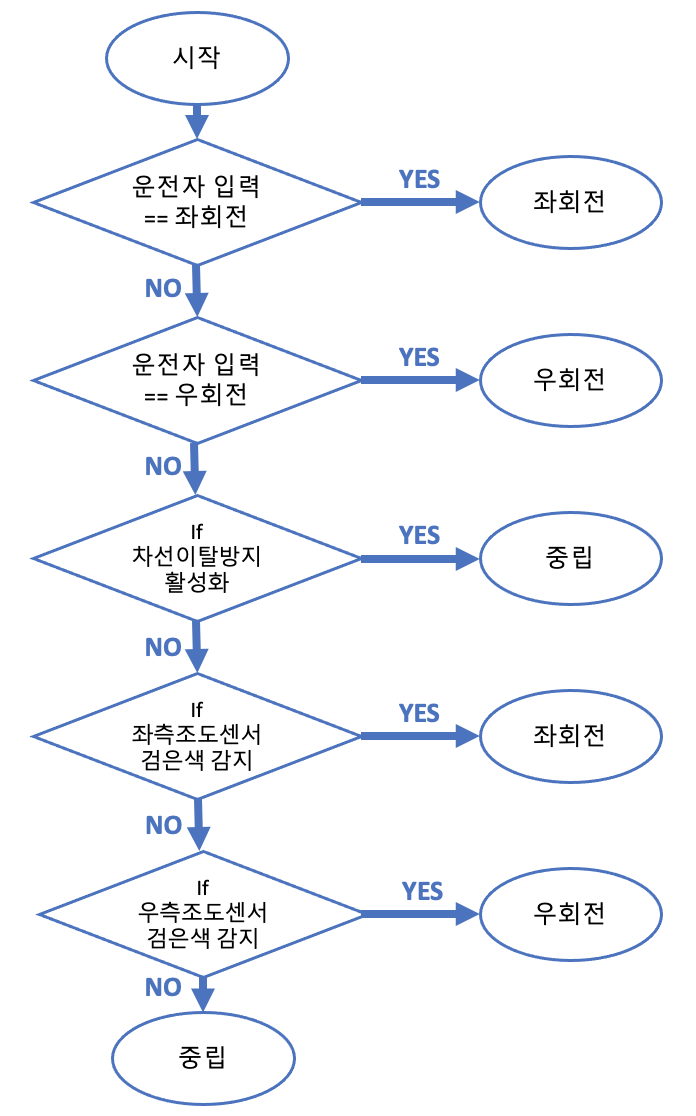
조향장치는 앞에서 조향장치의 각도를 변경하여 차량의 진행각도를 바꿀 수 있음을 알았는데 앞차의 기능에서 조향장치의 각도에 영향을 주는 요소는 다음과 같다.

* 운전자의 핸드폰 앱에서의 좌회전, 우회전 입력여부
* 추가기능인 차선이탈방지를 위한 조도센서 측정값

위의 두 가지 요소 중 우선순위가 높은것은 운전자의 입력이다. 차선이탈 방지모드는 차선이탈 방지만 담당하기 때문에 운전자가 차선변경을 원하거나 교차로를 대처할 순 없기 때문에 운전자의 입력을 우선순위가 높도록 하였다. 만약 운전자의 입력이 중립이고 차선이탈방지모드가 활성화된 경우 조도센서의 측정에 따라 조향장치의 방향이 변경될 것이다. 그전에 우리는 차선이탈방지모드를 다음과 같은 도로환경에서 작동할 수 있도록 설계하고자 한다.

* 차선은 흰색 바탕의 땅에 검은색 선으로 그려진다.
* 차선의 너비는 앞차의 양쪽 조도센서간의 간격보다 좁아야 한다.

위와 같은 환경에서 앞차는 조도센서에 검은색이 감지되는 쪽으로 방향을 틀어 차선이탈을 방지할 수 있을 것이다. 즉 조향장치의 방향은 다음과 같은 순서를 통해 결정된다.



4.4 뒷차 기능 분석

4.4.1 선행 차량 식별

뒷차는 선행 차량을 식별하고 차량 사이의 거리를 측정할 수 있어야 한다. 이를 위해 두 개의 초음파 센서를 이용하였다. 두 센서를 차량의 앞에 일정 간격을 두고 설치하여 전방의 사물을 감지할 수 있고 그 거리를 알 수 있다. 또한 서로 다른 두 센서의 측정값을 통해서 선행 차량의 좌 우 이동 뱡항을 알 수 있다. 하지만 센서를 통해 앞 차를 식별하는 과정에서 입력 받은 센서의 값을 그대로 사용하면 센서에 노이즈가 들어오거나 다른 센서와의 간섭이 일어나 차량이 앞 차와의 거리를 제대로 인식하지 못해 추돌, 정지하는 문제가 생기게 됐다. 이런 문제점을 해결하기 위해 첫번째로 앞차량을 식별하는 두 센서 사이의 간섭이 제일 적은 거리를 특정 하였다. 이 거리값을 안전 거리로 설정하여 센서 인식률을 제일 높은 상태로 유지하도록 하였다. 두번째로 노이즈를 최대한 식별하여 차량 주행에 영향을 주지 않도록 해야했다. 이 과정에서 노이즈를 필터링하는 알고리즘이 필요함을 느꼈고, 노이즈를 배열에 담아 평균값을 구하거나 중앙값을 구하는 방법을 생각했다. 센서에 들어오는 노이즈는 255로 들어오게 되어있고 앞 차와의 일정거리와 큰 차이가 생기기에 배열의 크기가 매우 크지 않으면 평균값을 구하는 알고리즘은 큰 효과가 없다는 것을 확인하였다. 따라서 배열의 적절한 크기를 유지하면서 노이즈를 식별할 수 있는 중앙값 필터 알고리즘을 사용하여 앞 차와의 거리를 구하였다.

4.4.2 전진, 후진 및 정지

뒷차는 선행 차량 식별 기능에서 측정된 앞차와의 거리를 통해서 전진 및 후진을 할 수 있어야 한다. 이를 위해 두 개의 모터를 이용하여 차량 후륜 구동을 구현하였다. 그러나 앞차량의 경우 고속 및 저속으로 이동 속도가 변하기 때문에 뒷차량은 앞차량과의 거리 차이에 따라 그 속도를 조절하여 안정적으로 앞차량과의 거리를 유지할 수 있어야 한다. 또한 앞차량의 움직임이 없는 경우 그리고 앞차량과 뒷차량 사이의 거리가 미리 정한 안전 거리에 들어오는 경우, 뒷차량은 이동을 정지해야할 필요가 있다. 그래서 입력된 센서값을 구간으로 나누어 그에 따라 뒷차를 고속 전진, 저속 전진, 정지, 후진으로 주행 할 수있게 하였다.

이 과정에서 차량 전진시 모터에 같은 값을 입력하여도 앞 차량의 속력보다 뒷차의 고속 주행보다 느리면 차량간 거리가 계속 벌어지게 되어 앞 차를 식별하지 못하는 경우가 생겼고 반대로 뒷차의 저속 주행보다 속도가 빠르면 차량이 자주 멈칫하는 경우가 생겼는데 이런 경우가 발생하는 이유는 차량의 배터리의 잔량에 따라 모터의 속도가 변하였음을 알아내었다. 그래서 모든 경우를 배터리가 완충 되었음을 가정하고 기능을 구상 하였다.

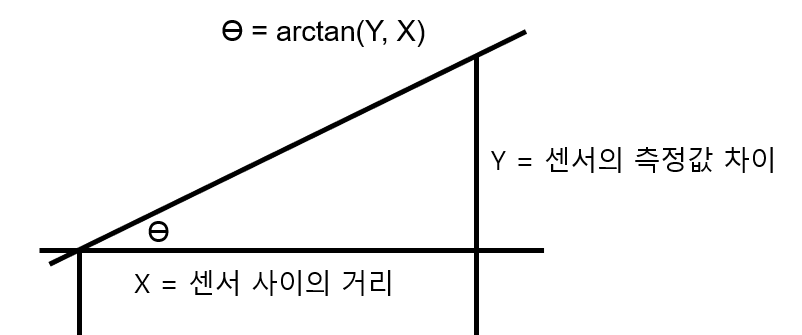
다음으로 차량간 거리가 빠르게 좁혀질 때 센서가 값을 받고 정지를 시키기 전에 차량이 추돌하는 경우가 생겼는데 이런 상황을 방지하기 위해 차량 간 안전거리의 거리를 주행 테스트를 하며 지속적으로 변화를 주어 가장 이상적인 정지 구간을 찾고, 센서를 입력 받는 주기를 빠르게 하며 기능구상을 하였다.

4.4.3 좌우 방향 전환

뒷차는 앞차의 방향 전환에 따라서 차량 이동 방향을 결정 짓게 된다. 결국 앞차와 동일한 스티어링 기능이 필요하다고 생각하였다. 그러나 사용자의 입력에 따라 동작하는 앞차와는 다르게 뒷차는 4.4.1에서 측정한 데이터를 이용하여 방향을 전환해야할 필요가 있다. 이 과정을 구현하면서 두 가지의 문제가 발생하였다. 첫번째로는 좌우 방향 전환을 구현함에 있어 단순히 모터를 회전시키는 것은 적합하지 않았다. 모터는 회전 명령을 받으면 계속 돌아가기 때문에 이를 제어하지 않으면 차량이 망가지는 상황이 발생할 수 있다. 따라서 회전의 최대각을 지정해주고 최대각 이상으로 모터가 회전하지 않도록 제어해줘야할 필요가 있다. 다행이 모터의 현재 회전각을 알 수 있는 API가 제공되어 있기 때문에 이를 이용해서 해당 문제점을 해결할 수 있었다. 두번째 문제는 앞차를 안정적으로 따라가려면 모터의 회전각을 세밀하게 제어해줘야할 필요가 있었다. 단순히 좌 우 센서 입력값의 차이로 좌우 회전 모터를 움직이면 모터의 회전 반경보다 센서 입력값의 차이 값이 절대적으로 작아 세밀한 조정이 힘들었다. 이를 해결하기 위해 Arctan 함수(그림 4-3)를 사용하였다. 해당 함수에 차량에 설치된 두 센서 사이의 거리, 두 센서 측정값의 차이를 인자로 넘겨주어, 엇각의 크기를 얻을 수 있다. 이 각은 Radian 값을 반환하는 반면에 모터의 회전각 API 는 ‘도’ 단위로 값을 반환한다. 따라서 Radian 값에 ( 180 / PI ) 를 곱해주어 단위를 맞춰주었다. 이렇게 구한 각도 만큼만 모터를 회전시켜 뒷차량 스티어링의 민감도를 높일 수 있었다.

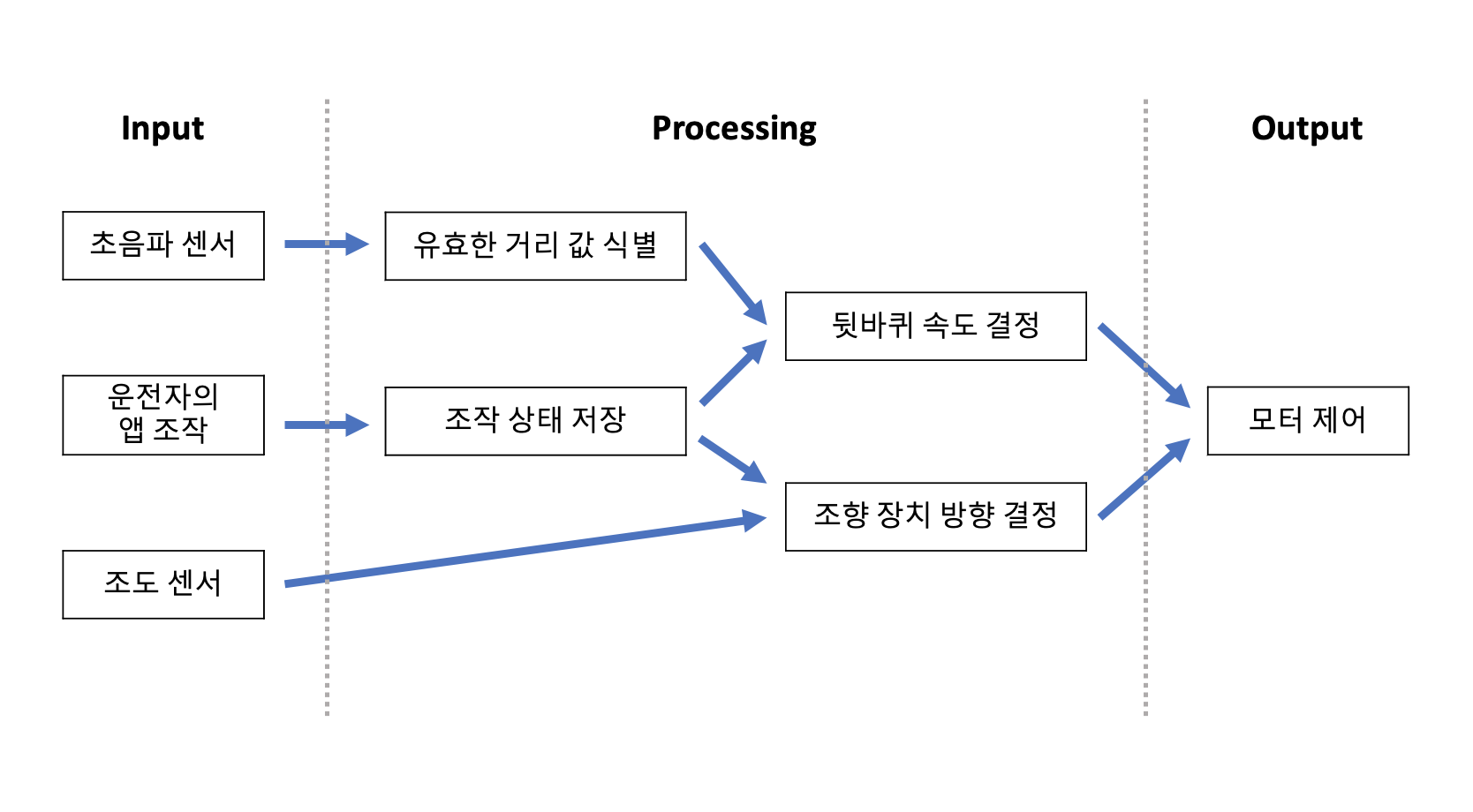
위의 두가지 문제를 해결하여 뒷차가 앞차의 방향을 따라 갈 수 있게 만들었지만 차량이 따라갈 때 방향 전환이 부드럽지 못하고 비틀비틀 거리며 따라가는 문제점이 생겼다. 첫번째로 한 고민은 센서끼리의 간섭이 심해 입력값을 제대로 입력받지 못한다고 생각하여 센서 위치를 조금 더 넓게하여 구상했으나 문제 해결엔 도움을 주지 못하였다. 두번째로 스티어링을 계산한 각도보다 미리 정지하게 하여 큰 회전을 작게 줄였으나 계산한 각도가 작으면 반응을 제대로 하지 못해 가장 민감해야할 스티어링의 목적에 맞지 않다고 생각했다. 그리고 마지막으로 앞차와의 각도는 실시간으로 빠르게 변화하는데 이를 센서로 빠르게 읽어들이지 못하는 문제라 생각하였다. 따라서 센서로 받아들이는 작업을 더욱 빠르게 만들고, 우선적으로 해서 결국 스티어링의 민감도를 앞차와 비슷하게 구상할 수 있었다.

(그림 4-3)



5. 설계

5.1 앞차 설계

앞차의 기능 분석에서 수행한 분석을 토대로 동작을 결정짓는 요소들을 식별했고 각 요소들에 영향을 주는 입력들과 입력의 출처를 알았기 때문에 다음과 같은 단계로 실제 모터 제어까지 도달할 수 있을 것으로 생각했다.

방향결정 단계

5.1.1 앞차 TASK 식별

위의 단계들은 3개의 TASK에 나누어 개별적으로 수행되도록 구상하였으며 각각의 TASK와 TASK가 담당하는 단계는 다음과 같다.

BtReceiver : 운전자의 앱 조작 -> 조작 상태 저장 -> 뒷바퀴 속도 결정

DistSensing : 초음파 센서를 통한 거리 측정 -> 유효한 거리 값 식별 -> 뒷바퀴 속도 결정

Operator : 조도센서를 통한 도로의 명암도 측정 -> 조향장치 방향 결정 -> 모터제어

추가로 블루투스가 연결되지 않은 초기상태에 지속적으로 연결을 시도하는 IdleTask가 존재한다.

각각의 TASK의 우선순위와 사용하는 Event, Resource등의 요소는 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TASK | Priority | Event | Resource | Autostart |
| BtReceiver | 2 | X | SpeedResource | X |
| DistSensing | 3 | SonicActivate | SpeedResource | O |
| Operator | 4 | X | X | X |
| IdleTask | 1 | X | X | O |

그리고 위의 TASK들을 제어하는 Alarm은 아래 표와 같이 세가지가 있으며 각각의 Action과 Parameter는 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| ALARM | 주기 | ACTION | PARAMETER |
| BtAlarm | 5 | ACTIVATETASK | TASK = BtReceiver |
| MotorAlarm | 5 | ACTIVATETASK | TASK = Operator |
| SonicAlarm | 50 | SETEVENT | TASK = DistSensing,  Event = SonicActivate |

5.1.2 BtReceiver

주요 특징 : Basic Task, priority = 2, Resource = SpeedResource, AutoStart = False, BtAlarm에 의해 5ms주기로 Activate 됨

BtRecevier Task는 핸드폰에서 전송되는 블루투스 패킷을 읽어 전역변수에 사용될 수 있는 형태로 가공하여 저장하는 Task이다. 이 때 정보 가공의 예로는 브레이크모드는 블루투스 패킷에선 급제동 모드 : 1, 서서히 제동 : 2로 수신되는데 nxt\_motor\_set\_speed() API에서는 brake flag로 급제동 모드 : 1, 서서히 제동 : 0으로 입력해야 하기 때문에 블루투스 패킷에서의 입력을 API에서 사용하는 값으로 변경하는 것을 정보 가공으로 표현했다. 또 운전자가 브레이크 작동을 입력한 경우 DistSensing에서 velocity를 담당하긴 하지만 velocity가 변경되는데 걸리는 시간이 오래 걸리고 브레이크는 즉시 반영 돼야 하는 경우가 많기 때문에 브레이크를 작동해야 하는 경우 BtReceiver에서 직접 velocity값을 0으로 변경한다. 때문에 DistSensing과 같은 전역변수를 수정하기 때문에 race condition이 발생하지 않도록 SpeedResource라는 Resource를 정의하여 velocity수정 시 get 하도록 한다. Basic Task로 설계되어 알람에 의해 5ms주기로 자주 호출되기 때문에 AutoStart는 필요하지 않아 False를 반영했으며 거리 측정을 방해하지 않도록 priority는 실제 차량 제어를 하는 Task중 가장 낮은 2로 설정했다.

5.1.3 DistSensing

주요 특징 : Extended Task, priority = 3, Resource = SpeedResource, AutoStart = True, 초음파 센서 측정 전 SonicActivate라는 Event를 wait함. SonicActivate는 SonicAlarm에 의해 50ms주기로 Set되어 초음파 센서를 이용한 측정간의 최소시간간격을 보장한다.

DistSensing은 충돌방지를 위해 거리를 수회 측정하고 측정된 거리 중 유효한 거리값을 식별해 유효하게 측정된 거리와 저장된 운전자의 입력을 바탕으로 속도를 결정하기 까지의 단계를 담당하는 TASK이다. 한 TASK에서 수회의 측정을 담당하기 때문에 그 측정간의 시간간격을 Event를 wait하고 Alarm이 50ms주기로 Event를 Set하여 보장하도록 설계했다.

속도는 분석에서 정한 과정을 통해 계산되며 velocity는 BtReceiver와 같이 수정하는 전역변수 이므로 역시 Resource를 사용한다.

5.1.4 Operator

주요특징 : Basic Task, priority = 4, AutoStart = False, MotorAlarm에 의해 5ms주기로 Activate됨

저장된 자동차 동작 관련된 전역변수에 따라 실제로 모터에 명령을 내리는 TASK이다. 뒷바퀴 모터들을 조작할 땐 속력엔 항상 전역변수 velocity를 사용하며 브레이크 모드의 경우 속력이 0이 아닐 때 천천히 멈춤인 0이 적용될 경우 비정상적인 작동을 하는 경우가 관찰되어 속력이 0이 될 때만 brake flag를 바꿔 적용하고 속력이 있을 땐 항상 1을 brake flag로 주어 문제를 해결했다.

좌, 우회전의 경우 조향 장치의 각도를 확인하여 좌, 우회전 또는 차선이탈 방지 기능이 작동하는 경우 일정 각도가 될 때까지 조향장치 모터를 회전시키고 각도가 넘어갈 시 모터를 멈추며 조향장치의 방향이 입력되지 않는 상태에선 각도가 0이 되도록 조향장치의 모터를 조절한다.

5.2 뒷차 설계

5.2.1 Task 설계

5.2.1.1 sensing ( basic )

* Basic task로 구현되었으며 sensing alarm에 의해서 50ms 의 주기로 호출 된다. 두 개의 초음파 센서를 이용하여 두 차량 간의 거리를 측정하는 것이 주된 목적이며 측정된 데이터를 좌/우 버퍼에 저장한 뒤 Quick sort 알고리즘을 이용하여 정렬한다. 이를 통해 노이즈를 제거할 수 있으며 최종적으로 중간값을 선택하여 좌/우 센서의 측정값을 확정한다. 두 측정값을 이용하여 조건문을 통해 SetEvent를 하고 조건에 맞는 Task를 Ready state로 전환한다. 초음파 센서의 데이터가 저장되는 버퍼는 critical section에 해당된다. 따라서 Resource를 이용해서 race condition을 방지하였다.

5.2.1.2 drive ( Extended)

* 지속적으로 호출되지 않고 특정 상황에서만 작동해야하기 때문에 Extended task로 구현되었으며 sensing task에서 발생한 DREvent를 통해서 Ready state로 전환된다. 해당 task는 후륜 구동에 필요한 두 개의 모터를 (-) 방향으로 회전시켜 차량이 앞으로 이동하게 한다. 회전을 마친 후 DREvent를 반환하고 다시 Wait 상태에 들어간다.

5.2.1.3 back (Extended)

* drive task 와 동일한 이유로 Extended task로 구현되었으며 sensing task에서 발생한 BAEvent 를 통해서 Ready state로 전환된다. 해당 task는 후륜 구동에 필요한 두 개의 모터를 ( + ) 방향으로 회전시켜 차량이 뒤로 이동하게 한다. 그 후 BAEvent를 반환하고 다시 Wait 상태에 들어간다.

5.2.1.4 stop (Extended)

* 앞선 두 Task 와 마찬가지로 Extended task로 구현되었으며 sensing task에서 발생한 STEvent를 통해서 Ready state로 전환된다. 해당 task는 후륜 구동에 필요한 두 개의 모터를 정지시킴으로서 차량이 멈추게 만든다. 그 후 STEvent를 반환하고 다시 Wait 상태에 들어간다.

5.2.1.5 steering (basic)

* 좌우 방향 전환은 선행 차량을 따라감에 있어서 가장 민감하게 반응해야하는 기능이다. 4.4.3 에서 서술했던 문제를 해결하기 위해 EVENT처리를 위한 함수를 사용하지 않는 basic task로 선언하였고, steering\_alarm 을 이용해서 15ms 주기로 호출되도록 하였다. 이를 통해 steering 의 반응속도 문제를 해결할 수 있었다. 또한 모터의 회전각을 결정하는 Arctan 연산 부분도 해당 task 에서 처리하도록 구현하였다.

5.2.2 Priority

steering task 의 경우 본 프로젝트의 목적을 구현하기 위해서 가장 민감하게 반응해야하는 task 라고 생각하였다. 따라서 5 개의 task 중 가장 높은 priority 인 5를 부여하였다. 또한 주행 중 안전거리 확보를 위해 stop task 에 그 다음 priority 인 4를 부여하였고, back, drive 순서로 3, 2 의 priority를 설정하였다.

5.2.3 Event

Event는 Extended Task로 구현된 drive, back, stop 을 ready state로 전환하기 위해서 사용하였다.

사용한 Event는 다음과 같다.

1. DREvent - sensing Task에서 양쪽 초음파 센서를 통해 입력된 값이 일정거리 이상이면 SetEvent 한다.
2. BAEvent - sensing Task에서 양쪽 초음파 센서를 통해 입력된 값이 일정거리 미만이면 SetEvent한다.
3. STEvent - sensing Task에서 양쪽 초음파 센서를 통해 입력된 값이 일정 구간으로 들어오거나 노이즈 값이 들어오면 SetEvent한다.

위와 같이 Event를 조건에 맞게 설정하여 다양한 상황에서 그에 맞는 주행을 하도록 설계하였다.

5.2.4 Resource

Critical section 으로 식별되는 자료 구조는 l\_buffer, r\_buffer , counter 총 세 개 이다. 따라서 critical section 에 접근하는 sensing task 는 race condition 을 방지할 수 있도록 resource를 활용하여 구현하였다. 다른 Task 들은

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TASK | Priority | Event | Resource | Autostart |
| sensing | 1 | X | sen\_resource | X |
| drive | 2 | O | X | O |
| back | 3 | O | X | O |
| stop | 4 | O | X | O |
| steering | 5 | X | X | X |

5.2.5 Alarm 설계

Alarm 은 시스템 상에서 동작하는 Clock 을 통해 주기적으로 Task를 Activate 시킬 수 있는 기능이다. 이러한 Alarm 을 basic task가 주기적으로 호출되도록 하기 위해 사용하였다. 또한 steering\_alarm 이 sensing\_alarm 보다 더 빠른 주기로 동작하도록 설정하였다. 이는 Priority 를 steering에 제일 높게 부여한 것과 동일한 의미로 해당 task 가 구현하는 동작이 민감하게 반응하도록 만들기 위해서 설계하였다.

|  |  |  |  |
| --- | --- | --- | --- |
| ALARM | 주기 | ACTION | PARAMETER |
| sensing\_alarm | 50 | ACTIVATETASK | TASK = sensing |
| steering\_alarm | 15 | ACTIVATETASK | TASK = steering |

5.2.6 아쉬운 점

뒷차를 구현하며 코드상으로는 문제가 없어도 하드웨어의 결함이 있으면 생각한대로 코드가 제대로 돌아가지 않아 아쉬움.

앞차가 후진시에는 뒷차의 스티어링이 반대로 되어야 하는데 시간이 부족하여 이 부분을 정확하게 구현하지 못한점이 아쉬움.

steering을 처음에는 extended task로 구현하고 이 부분의 문제일거라 생각을 못하고 basic task로 변환을 너무 늦게한 점이 아쉬움.

sensing 부분 또한 우선순위를 높여서 구현을 했다면 스티어링이 더 민감하게 작동해서 어떤 돌발 상황에서도 잘 following 했을 것 같아 아쉬움.